WORD LEXICON REDUCTION BY CHARACTER SPOTTING

Guilevic, D.; Nishiwaki, D.; Yamada, K.

Published in:
EPRINTS-BOOK-TITLE

Word Lexicon Reduction by Character Spotting

Didier Guillevic, Daisuke Nishiwaki, Keiji Yamada
Computer & Communication Media Research
NEC Corporation
Kawasaki 216-8555, Japan

We describe a system, currently under development, to dynamically reduce a lexicon of city names, making use exclusively of the information found in a word image. Isolated characters are 'spotted' within the word. The recognition results on those isolated characters are then used to initialize a Hidden Markov Model (HMM) like module to dynamically reduce the lexicon.

Keywords: Word Lexicon Reduction, Character Spotting, Finite Automata, Hidden Markov Models (HMM), Multi Layer Perceptron (MLP).

1 Introduction

The logical scheme to dynamically reduce the city/street name lexicon in a typical postal sorting application is to use the results from the zip code recognition. In non postal applications, some people make use of their generic word recognizer to perform the lexicon reduction, albeit with less parameters to make it more time efficient, and/or by reducing the size of the feature vector fed to the system. Jianying Hu [1], for example, performs on-line handwritten word recognition using a 2 stage HMM based system. The first stage is responsible for the lexicon reduction and performs an N-best decoding. That module is similar to the full word recognition module but uses a simplified feature set. In order to further reduce the computational load, the length of the observation sequence is also down sampled. Simard [2], similarly, used down sampling of the feature vector to build an efficient classifier. In his digit recognition application, an input pattern is compared to thousands of prototypes. Since the distance computation on full size prototypes is too expensive, comparison between an input image and the prototypes is performed at increasing resolution, thus drastically reducing the computation time.

As mentioned above, the usual scheme for city name lexicon reduction in postal applications is to use the recognition results from the zip code along with a database of zip code to city correspondence to generate dynamically a reduced lexicon. In the current project, we make the assumption that either the zip code could not be located, was missing or was simply not recognizable. Therefore we are interested in city lexicon reduction solely by analyzing a city word input image. We are currently investigating performing that lexicon reduction by spotting a few characters in a given word image. The recognition

In: L.R.B. Schomaker and L.G. Vuurpijl (Eds.), Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, September 11-13 2000, Amsterdam, ISBN 90-76942-01-3, Nijmegen: International Unipen Foundation, pp 373-382

results of those spotted characters is then used to match against the entries of the lexicon. The flow chart of the proposed system is shown in Fig. 1.

In the first stage of this project, we limit ourselves to all upper case words. They represent approximately 50% of the images in our database of Finland city names. Additionally, it seems to be possible most of the time to extract isolated characters from an all upper case word. The module, which decides whether or not an incoming word is written in all upper case letters, makes use of the location of the reference lines (lower and upper baselines).

2 System Modules

2.1 Pre-Processing

As in any handwriting recognition system, the first process attempts to reduce the variability of the handwriting by normalizing the vertical slant (Fig. 2) as well as the baseline tilt. The slant is estimated using some Slanted Vertical Histogram scheme [3]. We use the position of the upper and lower baselines to decide whether we are dealing with an all upper case or a mixed case word.

2.2 Word Length Estimate

The word length histogram for the city names found in the Finnish lexicon is shown in Fig. 3. Using the count of stroke crossings within the main body of a word, we estimate the number of characters. That information is then used to perform a first lexicon reduction. This process usually only brings a 50% reduction of the lexicon size, as most of the city names have a length between 7 and 10 characters.

2.3 Character Spotting

The next step consists of extracting (spotting) some characters from within the word. The easiest scheme is to extract isolated characters that do not overlap vertically. Namely locating in the image the columns of background pixels as depicted in Fig 4. Then for each bitmap, we need to decide whether or not, it is likely to be an isolated character. This involves the design of heuristics such as the maximum allowable aspect ratio, or thresholds on the degree of complexity (number of end points, strokes, etc...) in a bitmap image. In Fig 4, the bitmaps corresponding to the letters 'I', 'M' and 'U' are selected as plausible isolated characters. The simple scheme mentioned above has its limitations when neighbouring characters even slightly overlap. Therefore we moved to the analysis of connected components, which involves defining heuristic for merging neighbouring contours, disregarding some contours as noise, etc... The move to contour analysis enables the detection of more isolated characters (Fig. 4).

Figure 1: **Flow chart of the lexicon reduction engine**

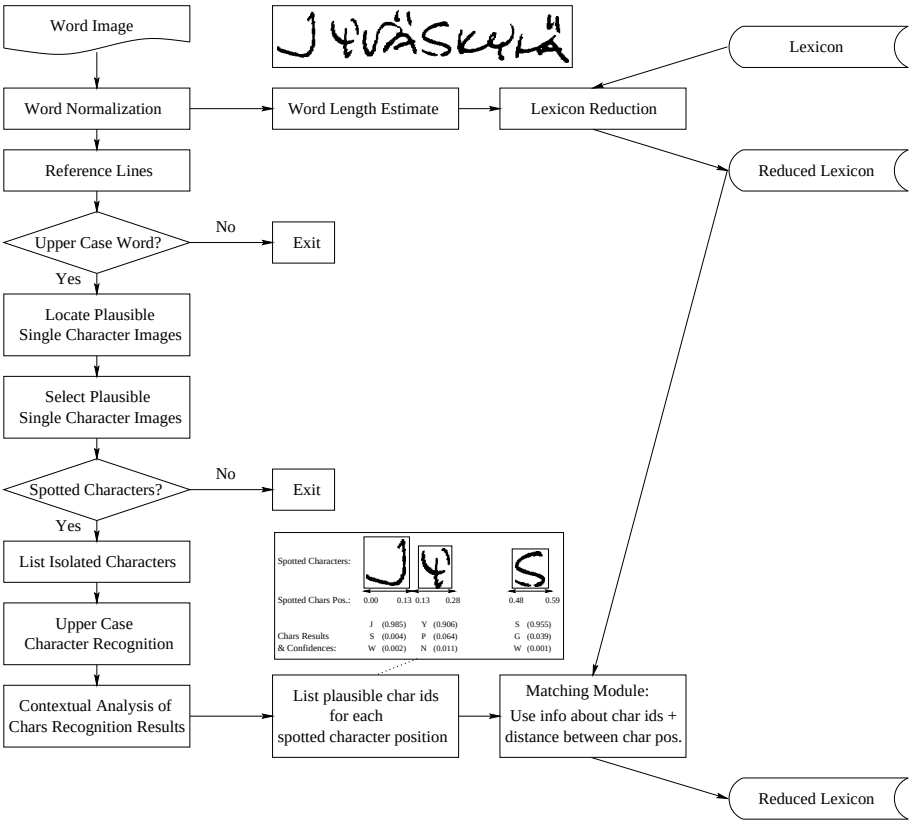


Figure 2: **Preprocessing: slant correction, position of reference lines**

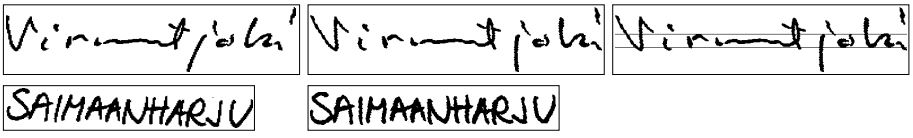


Figure 3: **Word length histogram for the Finland city name lexicon**

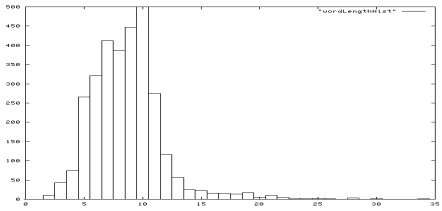
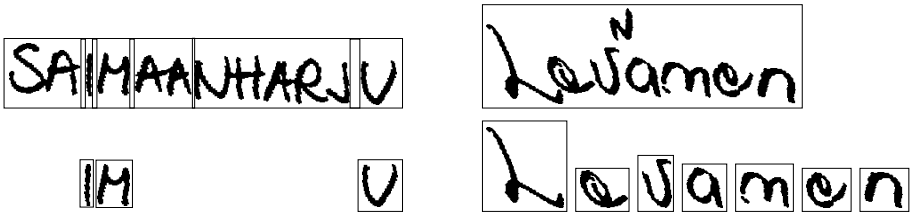
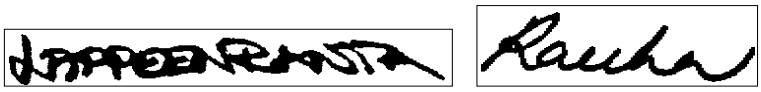


Figure 4: **Character Spotting: background columns / connected components**



If we are unable to locate some characters, then the image is rejected. Examples where the current scheme could not spot any characters are shown in Fig. 5. If we were to process those images, we would need to make use of some character segmentation module. As this would significantly increase the complexity, we could restrict ourselves to extracting only the first character in the word. Namely since the left boundary of the left most character in a word is known, that first character could probably be extracted with a relatively high confidence.

Figure 5: **Character Spotting: No character found**



2.4 Character Recognition

The spotted bitmaps are sent to the character recognizer illustrated in Fig. 6. The bitmaps are first scaled in binary to a standard size with the condition that the original aspect ratio be preserved. For each pixel, the distance to the nearest foreground pixel is computed. This generates 4 distance bitmaps, one for each of the east, west, south and north directions. After appropriate normalization, those bitmaps are fed to a standard multi layer perceptron (MLP) architecture. For a slight increase in performance, we code the outputs with Orthogonal Arrays (Hadamard Vectors) [4].

3 Matching

The next step consists of using the character recognition results (Fig. 7) to find those words in the lexicon that can be matched. One idea is to use the character results along with the relative position of the spotted characters to generate some kind of *finite automata*. We can then extract from the lexicon, those entries that can be generated by the automata, thus reducing the lexicon. Since we are interested in lexicon reduction, we can allow ourselves to consider the top few recognition results for each spotted characters. For the example given in Fig. 7, we would generate the following grammar:

```
? * {I,R,W} {M,H.U} ? * {U,V,N}
```

In the notation adopted, we assume that '?' matches exactly one character, while '*' matches any number of times (0 or more) any one character from the alphabet. In order to implement the grammar described above, we can make use of a Hidden Markov Model (HMM) module. The reader will recall that a HMM is nothing more than some enhanced implementation of a finite automata [5]. The discrete observation HMM generated for the example of Fig. 7 is shown in Fig. 8.

While we make use of an HMM module, we do not use it in the traditional way, and as such, the term HMM might not be entirely appropriate. We do not train the HMM (with Viterbi or Baum-Welch algorithms), but rather set the parameters 'by hand' to fit our intended purpose. Additionally, in our scheme, we do not force the emission probabilities within a given state to sum up to 1. The liberties we take will be described below.

The configuration of the Markov model is decided by the input word image. The number of symbols M is fixed and is equal to the size of our alphabet. For Finnish, the lexicon will be slightly larger than for English, as the alphabet needs to account for the Germanic characters with the umlaut (e.g. Ö). The number of states N of the discrete HMM is decided by the number of characters spotted as well as by their relative position. If 2 spotted characters are not

Figure 6: **Character Recognition Structure (MLP)**

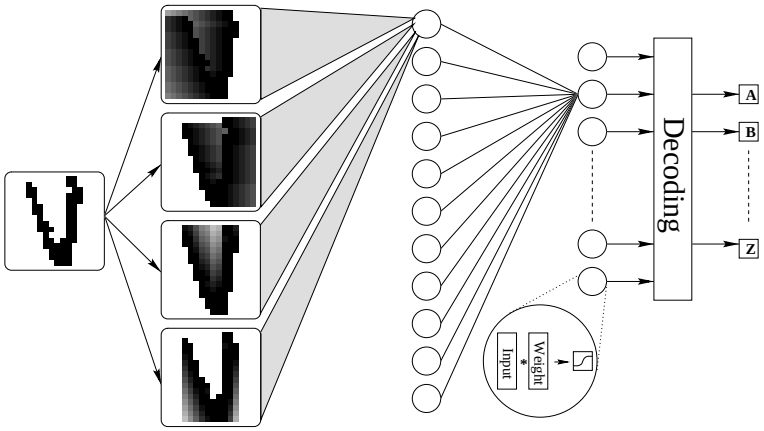
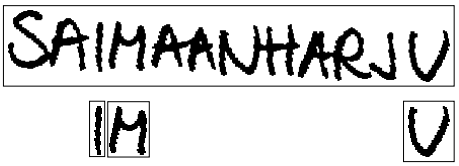
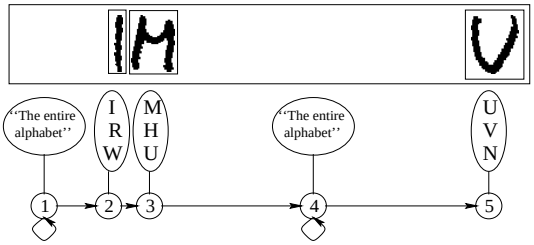


Figure 7: **Character Spotting and Recognition**



I 0.987016	M 0.873676	U 0.828351
R 0.004796	H 0.109930	V 0.140702
W 0.002206	U 0.007511	N 0.009280

Figure 8: **Finite Automata – Hidden Markov Model**



contiguous, then one state is added in between those 2 characters. In that additional state, any symbol from the alphabet is allowed. As an example, see the state added between the 'M' and the 'U' in Figure 8.

As mentioned briefly above, our HMM is not trained in the conventional meaning; rather the values of the matrices A (transition probabilities) and B (emission probabilities) are set by the designer. When, in a given state, any character from the alphabet is permitted, the probability of each character will be identical. On the other hand, when like in State 2 of Figure 8, we will expect the top 3 choices from the character recognition module to have a much higher probability than any other character in the alphabet.

The reader should note, that we are not constrained to have so-called "probabilities". The values in the matrices A and B can be whatever we wish them to be and should be chosen to fit our intended purpose. We chose some values such that, when a lexicon entry can indeed be generated by the automata, the HMM will give us a response of 0. On the other hand, if the entry matches all by one of the requirements, the HMM will output a response of -1, etc.... Figure 9 shows those cities from the Finland City Lexicon, for which the HMM gave a response of 0 (e.g. an exact match with the grammar). On that example, the lexicon has been reduced from $3,045$ to a mere 16 entries.

3.1 HMM Parameters

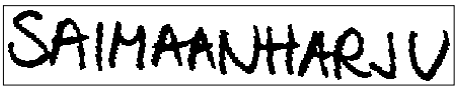
As briefly mentioned above, the HMM parameters are set to somehow implement a cost function. In the transition matrix A, permissible transitions are given a "probability" of 1, while others are set 0. For the emission probabilities (matrix B), a value of 1 is given to every permissible characters, while others are given a value of $exp(-1)$. Since an HMM typically outputs a *log* probability, each mismatch will add $log(exp(-1)) = -1$ to the final score. Fig. 10 illustrates an example where the correct lexicon entry is given a score of -1 because of the misrecognition of one character.

3.2 HMM Structure

If we wish to interpret the output of the HMM as a distance or cost measure, we need to slightly increase the complexity of the architecture in order to account for merged or split characters. In the example of Fig. 11, the correct lexicon entry is assigned a cost of -3 because the bitmap 'KI' is being misrepresented as a single character. A more meaningful distance would be a value of -1, representing one error. For that purpose, we change the representation of the basic HMM from that given in Fig. 8 to the one in Fig. 12.

Furthermore, to enable the correct processing of split character, we allow for the merging of adjacent characters as an alternate path in the HMM

Figure 9: **HMM Response: Cities matching the requirements**



Original Lexicon Size: 3045 Dynamic Lexicon Size : 16

HARMOINEN,MERIMASKU,KIUKAINEN,SÖÖRMARKKU,NOORMARKKU,URJALAN
KARHUKORSU, SÄRKIHARJU,SOMERHARJU,SAIMAANHARJU,YLIMARKKU,SÖDERUDDEN,
FURUHOLMEN,UIMAHARJU,JARHOINEN,VUOSTIMOJRV,HORMAKUMPU

Figure 10: **HMM Response: -1 (1 missed characters)**

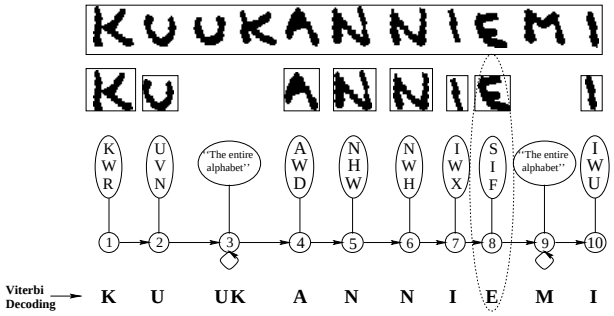


Figure 11: **HMM Response: -3: Error on connected characters**

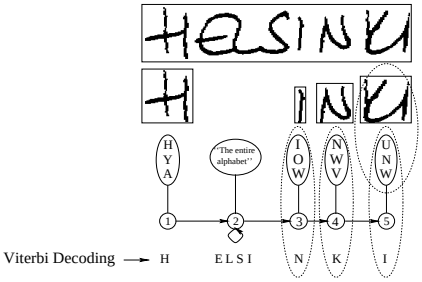


Figure 12: **HMM: Accommodating merged and split characters**

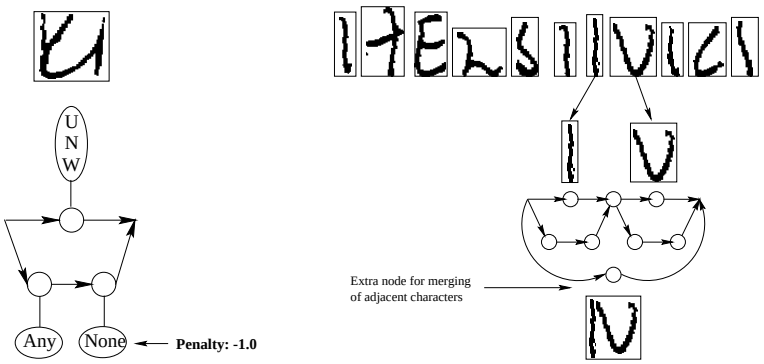
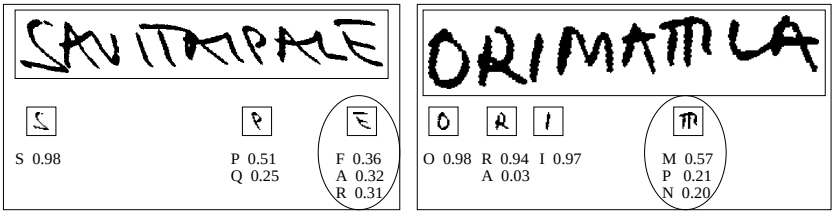


Figure 13: **Errors on all upper case word**



(Fig. 12). If the joined bitmaps form a plausible character, then the emission of the top few recognition results is permitted, else no character is permitted in that extra node, thus resulting in a penalty score of -1. Allowing for the merging of adjacent characters allows for images such as the one depicted in Fig. 12 to be successfully decoded by the HMM.

4 Experimental Results

The current testing is limited to 500 images taken randomly from a database of handwritten Finland addresses. Approximately half of those are written in all upper case, and the rest in mixed case. For about 7% of the images, the current scheme does not spot any characters, or none of the spotted characters meet the criteria for plausible isolated character. Those images are rejected by the present system.

Approximately 95% of the all upper case words are correctly processed by the current system, with the true label being in the dynamically reduced lexicon. The average reduced lexicon size is about 100 down from an initial value of 3,045 valid Finnish city names and abbreviations. Some of the errors are shown in Fig. 13. Currently, we only consider 'perfect' matches, therefore if a single character is not properly recognized within the top 3 choices of the character recognizer, then most likely the true id will not be generated by the HMM. The truth of the words shown in Fig. 13 have an HMM response of -1 as one of the spotted character is not recognized correctly. In the last example, the word 'ORIMATTILA' fails because the bitmap 'TTI' is being misjudged as being a single character.

5 Conclusion

We presented a system, currently under development, to dynamically reduce a lexicon of city or street names given some input word image. Isolated characters are spotted within the word image and the character recognition results are used to construct an HMM-like module. The latter is then used to dynamically reduce the city lexicon. The current scheme present some satisfactory results on all upper case word images. Further work will concentrate on building a better post-processing module for the character recognition results. This will include making use of the confidence values generated by the character classifier as well as taking into account various information such as similarity among certain classes, to prune and enhance the character results. From our experiments, we found out that an average of 3 correctly spotted and recognized characters per word is sufficient to significantly reduce the lexicon.

References

1. Jianying Hu, Sok Gek Lim, and Michael K. Brown. Writer independent on-line handwriting recognition using an HMM approach. *Pattern Recognition*, 33:133–147, 2000.
2. Patrice Y. Simard, Yann Le Cun, and John Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems*, volume 5, pages 50–58, 1993.
3. Didier Guillevic and Ching Y. Suen. Recognition of legal amounts on bank cheques. *Pattern Analysis and Applications*, 1(1), 1998.
4. Neil J.A. Sloane. A library of orthogonal arrays. "http://www.research.att.com/~njas/oadir/".
5. Didier Guillevic and Ching Y. Suen. HMM-KNN word recognition engine for bank cheque processing. In *Proceedings of the International Conference on Pattern Recognition*, pages 1526–1529, Brisbane, Australia, 1998.